Uma Ferramenta para Auxílio Didático no Ensino de Teoria da Computação

Yandre M. e G. da Costa[1], Rafael C. de Meneses[1], Flavio R. Uber[1,2]

[1]Departamento de Informática (DIN) – Universidade Estadual de Maringá (UEM)
Av. Colombo, 5790 – 87020-900 – Maringá – PR – Brasil.

[2]Faculdade de Filosofia, Ciências e Letras de Mandaguari (FAFIMAN).
Rua Renê Táccola, 152 – 86975-000 – Mandaguari – PR – Brasil.

{yandre, rcmeneses}@din.uem.br, flavio@fafiman.br

Abstract. *This paper describes a software environment for helping students and teachers on the creation of formal models studied on the theory of computation. It's a multiplatform environment that allows the user to perform syntactic tests with entry strings. This software tool offers the possibility of creating models useful for both regular language and context free language specification (important, for instance, for programming languages specification) and the Turing Machine model (useful for studies about computability). Thus, this software tool can support the learning of all language classes of the Chomsky Hierarchy.*

Resumo. *Este artigo discorre acerca de um ambiente de auxílio didático que permite a criação de modelos formais estudados em teoria da computação. Este ambiente é multiplataforma e permite realizar testes sintáticos sobre cadeias de entrada que são fornecidas pelo usuário. Esta ferramenta oferece a possibilidade de criar tanto modelos úteis para a especificação de linguagens regulares e livres de contexto (importantes, por exemplo, para a especificação de linguagens de programação), quanto a Máquina de Turing (modelo útil para estudos de computabilidade). Desta forma, a ferramenta pode apoiar o ensino de modelos relacionados a todas as classes da Hierarquia de Chomsky.*

1. Introdução

Conforme [McGettrick et al. 2004], conceitos matemáticos são importantes para prover especificações formais. A Teoria da Computação caracteriza-se como uma importante área de estudo da Ciência da Computação, pois nela estuda-se tanto fundamentos que descrevem o computador como um modelo matemático (máquinas universais), quanto modelos formais que permitem a especificação formal de linguagens. A descrição e a representação de linguagens formais são importantes para a Ciência da Computação, na medida em que as linguagens de programação precisam ser descritas formalmente. Desta forma, podemos verificar a importância de conceitos estudados em Teoria da Computação no processo de desenvolvimento de tradutores como compiladores e interpretadores.

Além do uso na especificação de linguagens, os modelos estudados em Teoria da Computação também têm outras importantes aplicações. A Máquina de Turing, por exemplo, caracteriza-se como a formalização de um procedimento efetivo [Menezes 1998]. Em outras palavras, pode-se dizer que a Máquina de Turing pode resolver qualquer problema que tenha solução algorítmica, podendo determinar quais problemas são computáveis e quais não são. Esses conceitos são extremamente importantes para o estudo dos limites da capacidade de solução de problemas pelo computador. Adicionalmente, modelos reconhecedores simples, como os Autômatos Finitos, também têm potencial de aplicação na modelagem de sistemas de estados finitos. Assim, percebe-se que os modelos formais compreendidos pelos estudos da Teoria da Computação são extremamente importantes para a Ciência da Computação estando inclusive relacionados às suas bases fundamentais.

Diante de tamanha importância da Teoria da Computação para a Ciência da Computação, e ainda considerando-se a grande dificuldade por parte dos alunos em assimilar estes modelos formais, já que além de muito abstratos, estes modelos possuem um forte caráter matemático, acredita-se que seja extremamente oportuno direcionar esforços em busca da melhoria do aprendizado destes conceitos por parte dos alunos de cursos de graduação. Neste sentido, este trabalho descreve o desenvolvimento do Software para a Criação e Teste de Modelos Formais (SCTMF). O ambiente é multiplataforma e permite que, além de modelar os formalismos estudados em Teoria da Computação, os usuários forneçam seqüências de símbolos (cadeias) como entrada, devolvendo o resultado (aceita/rejeita) da análise sintática realizada sobre a mesma e ainda o resultado final armazenado em estruturas de saída para modelos capazes de escrever durante o processamento de uma cadeia. Esta funcionalidade é particularmente útil no caso de modelagem de uma Máquina de Turing do tipo transdutora. Dentre os modelos formais já disponíveis no ambiente aqui descrito encontram-se: Autômato Finito Determinístico (AFD), Autômato Finito Não Determinístico (AFND), Autômato Finito com Movimento Vazio (AFMV), Expressão Regular (ER), Autômato com Pilha (AP), Gramática Livre de Contexto (GLC) e Máquina de Turing (MT). Embora existam muitas ferramentas disponíveis com propósitos similares aos desta, é importante observar que muitas delas não oferecem a possibilidade de se criar modelos que alcancem todas as classes de linguagem previstas na Hierarquia de Chomsky.

Este artigo encontra-se organizado da seguinte forma: na seção 2, serão apresentados os conceitos acerca de Linguagens Formais e Autômatos (LFA), incluindo uma descrição da Hierarquia de Chomsky e dos modelos formais que podem ser criados no ambiente; na seção 3, serão descritos detalhes do ambiente e do seu modo de funcionamento; na seção 4 serão relatadas as primeiras impressões obtidas sobre o uso do ambiente e os recursos disponíveis no mesmo serão comparados com os de outros ambientes similares; na seção 5 serão apresentados as conclusões e os trabalhos futuros.

2. Conceitos de Linguagens Formais e Autômatos

Nesta seção serão abordados os conceitos de LFA relacionados aos modelos disponíveis no ambiente aqui apresentado. Um dos pontos positivos deste ambiente reside no fato de que, através dele, é possível criar e realizar testes com modelos formais que contemplam todas as classes de linguagens previstas na Hierarquia de Chomsky [Menezes 1998]. Nas próximas subseções, serão apresentadas as definições matemáticas, descritas em [Menezes 1998], [Sipser 2005], [Vieira 2006] e [Hopcroft

2001], empregadas para a criação dos modelos no ambiente. Inicialmente, será apresentada uma breve descrição da Hierarquia de Chomsky.

2.1. Hierarquia de Chomsky

A Hierarquia de Chomsky, estabelecida por Noam Chomsky em 1956 [Chomsky 1956], classifica as linguagens em quatro classes diferentes, que são: Linguagens Enumeráveis Recursivamente (ou tipo 0), Linguagens Sensíveis ao Contexto (ou tipo 1), Linguagens Livres de Contexto (ou tipo 2) e Linguagens Regulares (ou tipo 3). Pode-se descrever uma relação de continência entre estas classes de linguagens onde a classe tipo 3 é um subconjunto da classe tipo 2, a classe tipo 2 é um subconjunto da classe tipo 1, e a classe tipo 1 é um subconjunto da classe tipo 0. A Figura 1 ilustra esta relação.

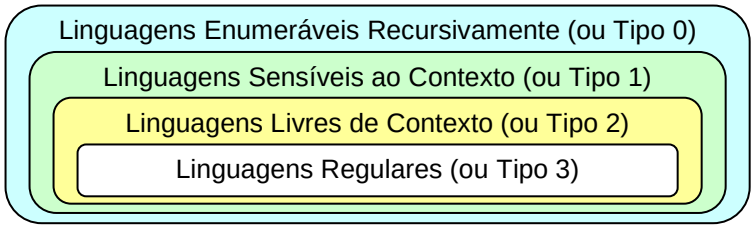

Figura 1. Relação hierárquica entre as classes de linguagens da Hierarquia de Chomsky

Cada modelo formal utilizado para a especificação de linguagens tem poder para alcançar uma das classes supracitadas. Deste modo, considerando-se as relações de continência descritas anteriormente, pode-se verificar que um formalismo com poder para representar uma linguagem enumerável recursivamente poderia representar qualquer linguagem presente em qualquer uma das outras categorias.

Embora um modelo para a representação de uma linguagem do tipo 0 seja suficiente para representar qualquer outra, é importante observar que quanto mais abrangente for a classe de linguagens maior será a complexidade do formalismo necessário para representá-la. Assim, a Hierarquia de Chomsky permite dimensionar melhor a ferramenta a ser utilizada para cada tipo de problema que se pretende resolver.

2.2. Modelos para Linguagens Regulares

Nesta seção serão descritas as definições matemáticas dos modelos com poder limitado à especificação de linguagens regulares que já estão implementados no ambiente descrito neste trabalho. Para esta classe de linguagem, existem três tipos de autômatos que podem ser criados, portanto três tipos de modelos de natureza reconhecedora de seqüências, além de expressão regular, um modelo de natureza geradora de seqüências. Levando-se em consideração que Gramáticas Regulares caracterizam um caso particular de Gramática Livre de Contexto, que será descrita na próxima seção, pode-se adicionar mais um modelo de natureza reconhecedora capaz de descrever linguagens regulares.

O Autômato Finito Determinístico (AFD) é definido como uma quíntupla $\langle \Sigma, S, \delta, S_0, F \rangle$, onde:

- Σ é um alfabeto de símbolos de entrada;
- S é o conjunto finito e não vazio de estados;
- δ é a função de transição, da forma $\delta: S \times \Sigma \rightarrow S$;
- S_0 é o estado inicial, $S_0 \in S$;
- F é o conjunto de estados finais, $F \subseteq S$.

O Autômato Finito Não Determinístico (AFND) é definido como uma quíntupla $\langle \Sigma, S, \delta, S_0, F \rangle$, onde:

- Σ é um alfabeto de símbolos de entrada;
- S é o conjunto finito e não vazio de estados;
- δ é a função de transição, da forma $\delta: S \times \Sigma \rightarrow \rho(S)$;
- S_0 é o conjunto de estados iniciais, finito e não vazio, $S_0 \subseteq S$;
- F é o conjunto de estados finais, $F \subseteq S$.

O Autômato Finito com Movimento Vazio (AFMV) é definido como uma quíntupla $\langle \Sigma, S, \delta, S_0, F \rangle$, onde:

- Σ é um alfabeto de símbolos de entrada;
- S é o conjunto finito e não vazio de estados;
- δ é a função de transição, da forma $\delta: S \times (\Sigma \cup \{\lambda\}) \rightarrow \rho(S)$;
- S_0 é o conjunto de estados iniciais, finito e não vazio, $S_0 \subseteq S$;
- F é o conjunto de estados finais, $F \subseteq S$.

Uma Expressão Regular (ER) sobre um alfabeto Σ é definida como segue:

- $\varnothing$ é uma ER e denota a linguagem vazia;
- λ é uma ER e denota a linguagem contendo exclusivamente a palavra vazia, ou seja, $\{\lambda\}$;
- Qualquer símbolo x pertencente a Σ é uma ER e denota a linguagem contendo a palavra x, ou seja, {x};
- Se r e s são ER´s e denotam as linguagens R e S, respectivamente, então:
  - (r+s) é ER e denota a linguagem $R \cup S$;
  - (rs) é ER e denota a linguagem $RS=\{uv|u \in R \text{ e } v \in S\}$;
  - (r^*) é ER e denota a linguagem R^*.

2.3. Modelos para Linguagens Livres de Contexto

Nesta seção serão descritas as definições matemáticas dos modelos com poder limitado à especificação de linguagens livres de contexto que já estão implementados no ambiente descrito neste trabalho. Para esta classe de linguagem, é possível criar Autômatos com Pilha, portanto um modelo de natureza reconhecedora de seqüências, e Gramáticas Livres de Contexto, um modelo de natureza geradora de seqüências.

O AP é definido como uma sêxtupla $\langle \Sigma, \Gamma, S, \delta, S_0, B \rangle$, onde:

- Σ é o alfabeto de entrada do AP;
- Γ é o alfabeto da pilha;
- S é o conjunto finito não vazio de estados do AP;
- δ é a função de transição de estados, $\delta: S \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow$ conjunto de subconjuntos finitos de $S \times \Gamma^*$;
- S_0 é o estado inicial, $S_0 \in S$;
- B é o símbolo da base da pilha, $B \in \Gamma$.

A GLC é definida como uma quádrupla $\langle V, T, P, S \rangle$, onde:

- **V** é um conjunto finito de símbolos não-terminais (ou variáveis);
- **T** é um conjunto finito de símbolos terminais disjunto de V;

- **P** é um conjunto finito de pares, denominados regras de produção tal que a primeira componente é um elemento do conjunto V e a segunda componente é palavra de $(V \cup T)^*$;
- **S** é um elemento de V, denominado símbolo inicial (ou de partida).

2.4. Máquina de Turing

Nesta seção será descrita a definição matemática utilizada para a implementação do módulo que permite a criação de Máquinas de Turing, modelo mais poderoso dentre os modelos matemáticos estudados em Teoria da Computação [Divério e Menezes 2000]. A Máquina de Turing tem natureza reconhecedora, e é bastante útil inclusive nos estudos relacionados a computabilidade, que investiga os limites da capacidade de solução de problemas pelos computadores. A seguir é descrita a definição matemática empregada na implementação do ambiente para a criação de Máquinas de Turing.

A MT é definida como uma óctupla $\langle \Sigma, S, \delta, S_0, F, V, \beta, ¤ \rangle$, onde:

- Σ é o alfabeto de símbolos de entrada;
- S é o conjunto de estados possíveis, o qual é finito;
- δ é o programa ou função de transição
  - $\delta: S \times (\Sigma \cup V \cup \{\beta, \langle\}) \rightarrow S \times (\Sigma \cup V \cup \{\beta, \langle\}) \times \{E, D\}$ a qual é uma função parcial;
- S_0 é o estado inicial da máquina, $S_0 \in S$;
- F é o conjunto de estados finais, $F \subset S$;
- V é o alfabeto auxiliar (pode ser vazio);
- β é o símbolo especial para células em branco;
- ¤ é o símbolo especial marcador de início da fita.

3. O ambiente SCTMF

Esta seção descreve como é o processo de criação de alguns modelos no ambiente SCTMF (disponível em http://myjavaserver.com/~cassolato). Para isto, foram escolhidos os modelos GLC e MT. Assim, serão envolvidos dois dos mais importantes e conhecidos modelos estudados em Teoria da Computação e que se diferenciam no que diz respeito à forma como analisam cadeias. A GLC tem natureza geradora enquanto a MT têm natureza reconhecedora. Adicionalmente, é válido ressaltar que estes modelos têm poder para representar linguagens de diferentes classes da Hierarquia de Chomsky. A GLC é capaz de especificar linguagens livres de contexto, enquanto a MT é capaz de especificar linguagens enumeráveis recursivamente. A fim de atender restrições de tamanho para a escrita deste artigo, os outros modelos disponíveis no ambiente (AFD, AFND, AFMV, ER, AP) não serão mostrados nesta seção. Observe que o conjunto completo de modelos oferecidos pela ferramenta permite a descrição de modelos que podem alcançar qualquer classe de linguagem descrita na Hierarquia de Chomsky (considerando que MT também cobre linguagens sensíveis ao contexto).

A Figura 2 mostra a interface principal do ambiente SCTMF. Observe que nesta tela o usuário, a partir de botões dispostos em uma barra localizada na parte superior, pode escolher qualquer um dos sete modelos formais disponibilizados de forma que possa iniciar a criação de um destes modelos. Observe ainda que, conforme ilustra a mesma figura, o usuário pode criar simultaneamente modelos de diferentes tipos.

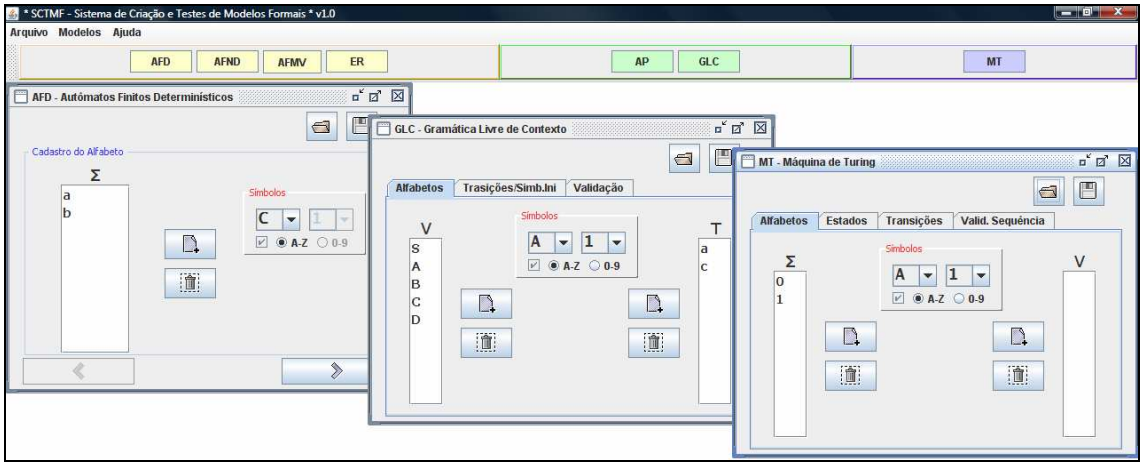

Figura 2. Interface principal do SCTMF

A seguir serão ilustradas as seqüências de passos para a criação de uma instância de cada um dos dois modelos supracitados. Inicialmente, será mostrada a criação de uma GLC.

A Figura 3 mostra a seqüência de janelas que aparecem para que o usuário descreva os componentes da quádrupla de uma GLC, conforme descrito na seção 2.3.

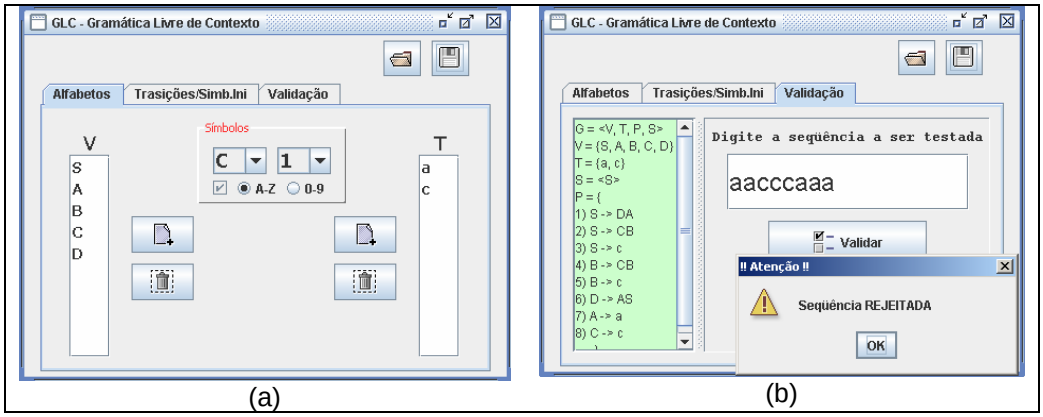

Figura 3. Passos para a criação de uma GLC no ambiente SCTMF

A Figura 3(a) mostra a janela onde o usuário pode definir os alfabetos de símbolos terminais T e de símbolos não terminais (ou variáveis) V, neste ponto existe uma restrição que só permite a criação de símbolos descritos por um único caractere, no caso dos símbolos terminais este caractere pode corresponder a uma letra ou a um dígito numérico. A partir de uma guia apresentada nesta mesma tela, o usuário pode estabelecer o símbolo inicial S e o conjunto de regras de produção da gramática. Na Figura 3(b) é ilustrada a janela que permite ao usuário inserir uma seqüência de símbolos e verificar se a mesma pode ser gerada ou não pelo modelo criado. Observe que nesta janela o usuário pode visualizar a descrição completa da quádrupla que ele definiu. A GLC criada neste exemplo reconhece a linguagem $\{a^n c^m a^n \mid n \geq 0 \wedge m \geq 0\}$. Note que, neste caso a cadeia de teste 'aacccaaa' não foi aceita por não fazer parte da linguagem, já que no seu prefixo apresenta uma seqüência de dois símbolos 'a' e no sufixo aparecem três. Para teste das cadeias submetidas na GLC foi utilizado o algoritmo de Cocke-Younger-Kasami, descrito em [Menezes 1998]. Observe ainda que, para alternar entre as diferentes etapas de criação da GLC, basta que o usuário

selecione, nas janelas mostradas na Figura 3, a guia correspondente a etapa de criação desejada.

Considerando a definição matemática da MT, descrita na seção 2.4, a Figura 4 mostra a seqüência de janelas para o usuário definir os elementos da MT.

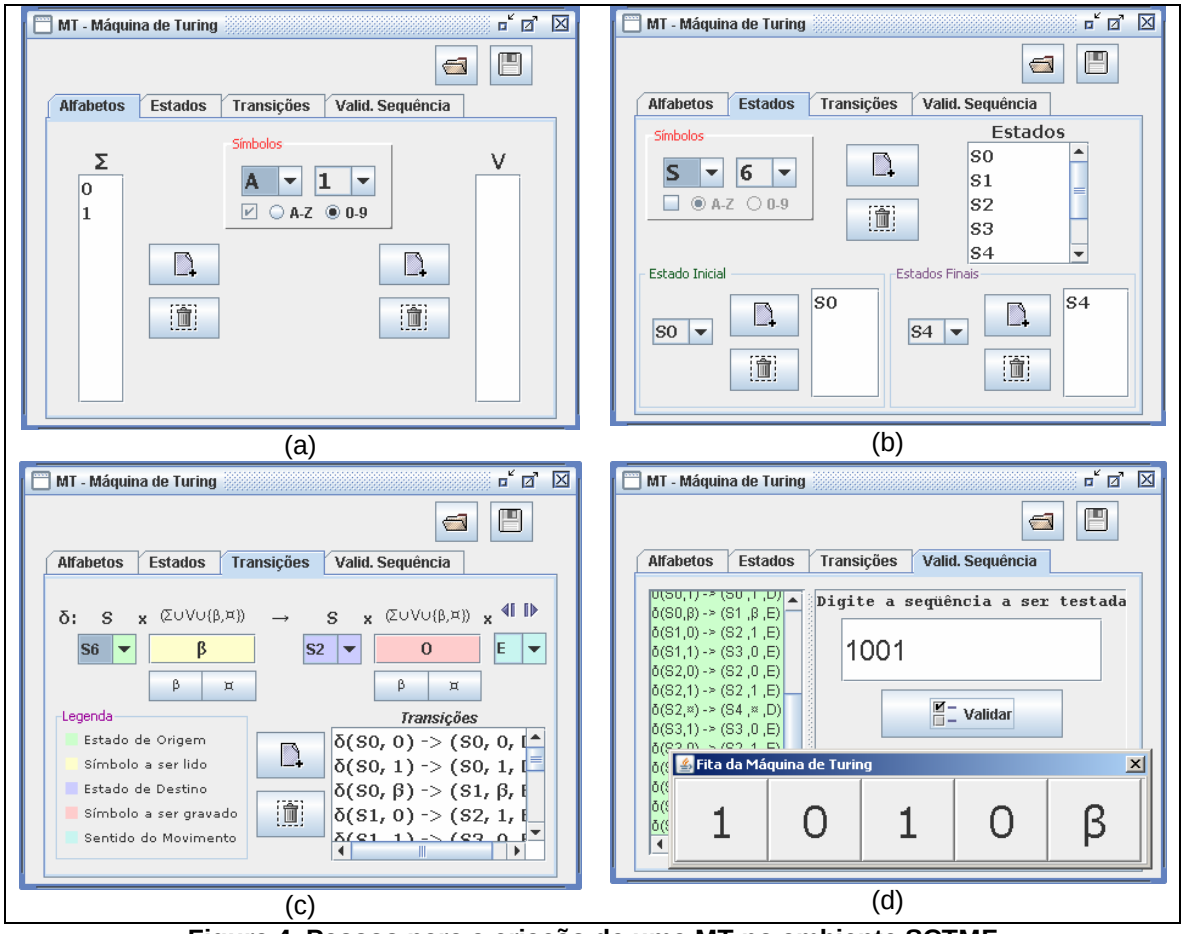

Figura 4. Passos para a criação de uma MT no ambiente SCTMF

A Figura 4(a) mostra a janela onde o usuário pode definir os alfabetos de símbolos Σ e V, neste ponto existe uma restrição que só permite a criação de símbolos descritos por um único caractere, no caso dos símbolos de Σ este caractere pode corresponder a uma letra ou a um dígito numérico. A Figura 4(b) mostra a janela onde o usuário pode estabelecer o conjunto de estados S, o estado inicial S_0 e o conjunto de estados finais F. Na Figura 4(c) é mostrada a janela que permite a criação das funções de transição δ da MT (definida neste caso como função parcial). Na Figura 4(d) é ilustrada a janela que permite ao usuário inserir uma seqüência de símbolos a ser processada pela MT. O usuário sempre recebe uma resposta do tipo aceita/rejeita em função da parada da MT com um estado final ativo ou não. Entretanto, em alguns casos são criadas máquinas do tipo transdutora [Vieira 2006], existindo, portanto, uma preocupação com o conteúdo final da fita da máquina. Por isso, além da resposta aceita/rejeita, o ambiente mostra o conteúdo da fita ao término do processamento da cadeia. A MT criada no exemplo descrito na Figura 4 é uma MT do tipo transdutora capaz de receber uma seqüência de dígitos que corresponda a um número binário e devolver, ao final, o valor correspondente ao da entrada incrementado em uma unidade. Note que, neste caso a seqüência fornecida na entrada foi ‘1001’ e ao término do

processamento o conteúdo da fita era ‘1010’. Nesta janela o usuário também pode visualizar a descrição completa da óctupla que ele definiu. Observe ainda que, assim como na GLC, para alternar entre as diferentes etapas de criação da MT, basta que o usuário selecione, nas janelas mostradas na Figura 4, a guia correspondente à etapa de criação desejada.

Todo modelo criado no SCTMF pode ser salvo pelo usuário, assim é possível que o usuário venha a utilizar o modelo num momento posterior e fazer alterações eventualmente necessárias.

4. Resultados obtidos com o uso do SCTMF

Avaliações empíricas realizadas em situações práticas que envolveram o uso do SCTMF em atividades de construção de modelos formais por alunos da disciplina de Teoria da Computação do curso de Bacharelado em Ciência da Computação da Universidade Estadual de Maringá, sugerem que quanto maior a complexidade do modelo a ser construído, maior tende a ser o auxílio prestado pela ferramenta. Acredita-se que isto aconteça pelo fato de que, em modelos menos complexos, a simples tarefa de criar o modelo no ambiente já seja responsável por boa parte do tempo gasto na atividade, o que desestimula o uso da ferramenta para a criação de modelos que consistem em problemas mais simples. Em modelos mais complexos, o tempo gasto na criação é diluído e, adicionalmente, os testes de verificação de seqüências contribuem mais em termos de auxiliar na identificação de eventuais erros cometidos na criação do modelo.

Embora não tenha sido feito nenhum levantamento ou avaliação sistematicamente organizado junto aos alunos a fim de avaliar o impacto do uso da ferramenta, observou-se que, de forma geral, há uma boa aceitação em relação ao uso da ferramenta, na medida em que o *feedback* imediato ajuda na verificação da correção dos modelos criados impactando significativamente nos resultados finais obtidos e na retenção do aprendizado dos conceitos explorados.

Os ensaios preliminares realizados nas aulas da disciplina apontam que a ferramenta tem forte potencial de uso nas seguintes situações:

- Uso como ferramenta para auxiliar didaticamente o professor na exemplificação de modelos formais e demonstração da dinâmica de funcionamento dos mesmos;
- Uso como ferramenta para auxiliar os alunos na construção de modelos formais em aulas práticas em laboratório;
- Uso como ferramenta distribuída pela internet para auxiliar os alunos na solução de exercícios de forma não presencial.

4.1. Ferramentas similares

Em razão das dificuldades encontradas no ensino de Teoria da Computação, não é novidade a existência de ferramentas com o propósito de proporcionar um ambiente de ensino mais produtivo tanto para o aluno quanto para o professor.

O que diferencia as ferramentas existentes são os modelos tratados por cada uma delas. A Tabela 1 sumariza importantes funcionalidades de algumas destas ferramentas. Através dela, pode-se visualizar algumas características em comum e outras que diferenciam tais ferramentas.

Tabela 1. Comparação das funcionalidades implementadas em cada ferramenta

	VAS	GAM	JFlap	Language Emulator	SCTMF
AFD	X	X	X	X	X
AFND	X	X	X	X	X
AFMV				X	X
ER					X
MT	X		X		X
Máquina de Mealy			X	X	
Máquina de Moore			X	X	
Transformação AFND → AFD	X	X		X	
ER			X	X	X
GR			X	X	X
Transformação AFD → ER				X	
Minimização de AFD		X		X	

Dentre as quatro ferramentas comparadas com a SCTMF, as ferramentas GAM [Jukemura et al. 2005] e Language Emulator [Vieira et al. 2003] foram desenvolvidas por pesquisadores brasileiros, ao contrário das ferramentas VAS [Bovet 2008] e a Jflap [Rodger 2007]. De modo geral, pode-se observar que existem ferramentas mais completas e outras com propósitos bem específicos com relação às funcionalidades implementadas. Com isso percebe-se que o grande diferencial da SCTMF é o fato de ser a única ferramenta nacional (e com interface em português) que possibilita a simulação de MT e também de outros modelos de natureza geradora de seqüências.

É válido ressaltar que a implementação das funcionalidades já disponíveis na SCTMF foi realizada por um aluno de graduação como trabalho de conclusão de curso. Por questões ligadas à limitação de tempo, algumas funcionalidades ainda não foram implementadas, mas a continuidade do trabalho de implementação será dada por outro aluno de graduação, garantindo que funcionalidades ainda não disponíveis venham a ser oferecidas pela ferramenta. Dentre estas funcionalidades, destaca-se a transformação entre modelos que possuem equivalência, como: AFND-AFMV-ER-AFD e AP-GLC.

5. Considerações finais

Dentre as funcionalidades disponibilizadas no SCTMF, a possibilidade de se criar Máquinas de Turing pode ser apontada como o principal destaque da ferramenta. Este fato diferencia a ferramenta da maioria das ferramentas similares e permite inclusive que nela sejam especificadas linguagens de todas as classes da Hierarquia de Chomsky. Adicionalmente, a ferramenta mostra ao usuário o conteúdo final armazenado na fita ao término de um processamento realizado pela MT. Isto é particularmente útil no caso de se modelar Máquinas de Turing do tipo transdutora.

O SCTMF também tem como ponto positivo o fato de ser uma ferramenta multiplataforma. Desenvolvido com Java Swing, pode ser utilizado em qualquer plataforma que suporta máquina virtual Java.

5.1. Trabalhos futuros

Dentre as funcionalidades que devem ser incluídas nos trabalhos futuros de desenvolvimento da ferramenta, encontram-se:

- Transformação entre modelos equivalentes, como AFD-AFND-AFMV-ER e AP-GLC;

- Otimização de AFD;
- Implementação de módulo que permita a criação de Autômato Linearmente Limitado, modelo que cobriria exatamente a classe de linguagens sensíveis ao contexto;
- Implementação de recursos que permitam a criação dos modelos reconhecedores visualmente, através da construção de diagramas graficamente.

Além disso, deverá ser adotada uma metodologia sistematicamente organizada a fim de avaliar a eficácia e a eficiência do ambiente como ferramenta para auxiliar os alunos na solução de problemas que envolvem a criação dos modelos formais. Conforme mencionado neste artigo, existem evidências de que a contribuição da ferramenta tende a ser maior em atividades que envolvam modelos mais complexos. Desta forma, esta avaliação depende de um espaço de tempo no período letivo suficiente para abranger o ensino dos diferentes modelos formais disponíveis na ferramenta.

Referências

Bovet, J. (2008) "VAS – Visual Automata Simulator", http://www.cs.usfca.edu/%7Ejbovet/vas.html, Acessado em Fevereiro de 2008.

Chomsky, Noam (1956) "Three Models for the Description of Language", em: IRE Transaction on Information Theory.

Divério, T. A. e Menezes, P. B. (2000) "Teoria da Computação – Máquinas Universais e Computabilidade", Sagra Luzzatto, 2ª. Edição.

Hopcroft, J. E., Motwani, R. e Ullman, J. D. (2001) "Introduction to Automata Theory Languages, and Computation", Addison Wesley, 2a. Edição.

Jukemura, A. S., Nascimento, H. A. D. do e Uchôa, J. Q. (2005) "GAM – Um Simulador para Auxiliar o Ensino de Linguagens Formais e de Autômatos", em: XIII Workshop sobre Educação em Computação – XXV Congresso da Sociedade Brasileira de Computação.

McGettrick, A., Boyle, R., Ibbett R., Lloyd, J., Lovegrove, G. e Mander, K. (2004) "Grand Chalenges in Computing – Education". The British Computer Society.

Menezes, P. B. (1998) "Linguagens Formais e Autômatos", Sagra Luzzatto, 2ª. Edição.

Rodger, S. H., Lim J. e Reading, S. (2007) "Increasing Interaction and Support in the Formal Language and Automata Theory Course", em: The 12th Annual Conference on Inovation and Thecnology in Computer Science Education (ITiCSE 2007).

Sipser, M. (2007) "Introduction to the Theory of Computation", Course Technology, 2ª. Edição.

Vieira, L. F. M., Vieira, M. A. M. e Vieira, N. J. (2003) "Language Emulator, uma ferramenta de auxílio no ensino de Teoria da Computação", em: XIII Workshop sobre Educação em Computação – XXV Congresso da Sociedade Brasileira de Computação.

Vieira, N. J. (2006) "Introdução aos Fundamentos da Computação – Linguagens e Máquinas", Thomson Learning Edições, 1ª. Edição.